小组成员

姓名	学号	分工	自评分
甘顺豪	202330450431	完成主要函数实现	96
赖裕安	202330450821	完成部分框架	93
李志宏	202330450991	参与思路构建	90
@幸福记号	未知	参与思路构建	
@L-焕	未知	参与思路构建	

算法流程

1、用节约值最低优先合并路径算法（概率跳过最优）生成 50 个初始解。
2、每次迭代前自适应地调整惩罚系数。
3、每次迭代前根据适应度排序，并保留父代前五个优秀解（伴随随机变异）。
4、在父代中根据交叉算子生成解并经历随机变异后作为新解。
5、将前五个优秀解和 45 个新解作为新父代，迭代循环 1000 次。
6、迭代 1000 次后检查解的合法性，将最优秀的合法解作为最优解。
7、若无合法解保证评估次数小于 49800 的情况下迭代修补和变异至得到最优合法解

伪代码

```
for (int i = 0; i < 50; i++){
    individual I(VNum - 1, CarNum, MaxLoad);
    I.initialize(G);
    I.repair_overload_routes(G);
    generation_1.push_back(I);
}
G.self_adaptation(generation_1,pid_p);
sort(generation_1.begin(), generation_1.end(), compareByE);
 for (int n = 1; n < 1000; n++) {
     //保留优秀样本
      for (int i = 0; i < 5; i++)generation_2.push_back(generation_1[i]);
     //增加交叉繁殖和变异后样本
     vector <individual>crosresult = move(crossover(generation_1, 45, G));
     for (individual I : crosresult){
         generation_2.push_back(I.mutate(G));
         if (generation_2.size() >= 50)break;
     }
     //准备下一代
     generation_1 = move(generation_2);
     G.self_adaptation(generation_1,pid_p);
     std::sort(generation_1.begin(), generation_1.end(), compareByE);
 }
```

```
return valid_best_answer;
```

参数设置

种群规模：50
迭代次数：1000
惩罚系数初始： p=500 * G.avg_dist / G.avg_q;
pc1 = 0.9, pc2 = 0.4, pm1 = 0.1, pm2 = 0.001;
适应度 e =routeΣ（ dist + ((demand > Q) ? G.p * (demand - Q) : 0))
变异概率 pc = (it->e < G.avg_e) ? (G.pc1 - (G.pc1 - G.pc2) * (G.avg_e - it->e) / (G.avg_e - G.min_e)) : G.pc1;
交叉概率 pm = (e < G.avg_e) ? (G.pm1 - (G.pm1 - G.pm2) * (G.avg_e - e) / (G.avg_e - G.min_e)) : G.pm1;

自适应参数调整算法

对当前一代个体的误差和有效性进行评估，找出最小的误差和平均误差，并计算无效个体的比例；根据当前一代个体的无效比例与目标比例之间的差异，计算出一个误差值；使用 PID 控制器根据计算出的误差来输出 delata_p,调整参数 p，使其朝着理想状态（即目标无效比例 target_invalid_rate = 0.2）靠近；对调整后的参数进行约束，确保其在一个预定的合理范围内，避免过大或过小，保证系统的稳定性。
评估当前代个体性能

误差统计：计算当前代所有个体的误差（如路径总距离、时间成本等），记录 最小误差 emin 与平均误差 eavg

无效个体判定：若个体的误差超过阈值或违反约束（如容量超限），标记为无效。统计 无效比例
定义目标无效比例
rtarget=0.2
计算误差值
Error=rinvalid−rtarget
PID 控制器动态调整参数

基础参数计算：
定义
min_output= avgdist/avgdemand
PID 参数设定：
Kp=7×min_output ， Ki=10×min_output， Kd=0（纯 PI 控制）
调整量约束：｜Δp｜≤8×min_output（避免突变）。
更新参数并施加全局约束

节约值最低优先合并路径算法

计算路径间节约值
遍历所有可能合并的路径对 ，计算合并后的“节约值”（例如：合并前两条路径的总成本 - 合并后新路径的成本）。

动态选择合并操作

将路径对按节约值 降序排列（高节约值优先）。

为避免局部最优，采用 概率化选择策略（0.9 的概率跳过），优先选择高节约值路径对，但允许次优合并以维持多样性。

合并路径并验证约束

依次尝试合并选中的路径对，若合并后的路径满足约束（如容量、时间窗），则保留新路径并移除原路径；否则跳过。

重复此过程直至无法合并或达到预设迭代次数。

补全缺失节点

合并完成后，检查未被覆盖的节点（未分配至任何路径的客户点）。

将缺失节点按 贪心策略 插入现有路径：

优先插入到 负载最低的可行路径（如剩余容量最大的车辆路径）；

若无法插入，创建新路径。

终止条件

所有节点均被分配且路径满足约束条件，或达到最大计算时间/迭代次数

变异算子：该函数用于对个体（解）进行变异操作。变异概率 `pm` 根据个体的适应度值 `e` 动态调整：当个体的适应度优于图的平均适应度时（`e < G.avg_e`），使用线性插值降低变异概率；否则使用基础概率 `G.pm1`。根据随机数判断是否触发变异，若触发则随机选择两种变异方式之一：

交换路径内客户：在同一路径中随机交换两个客户的位置。

跨路径迁移客户：将一个客户从某条路径迁移到另一条路径。

变异完成后，重新评估个体的适应度。

交叉算子：亲本大于 pc 的概率进行交叉，随机交换部分路径，并进行去重和补缺；